Preprint
UCRL-JC-142243

Supervised and Unsupervised Discretization Methods for Evolutionary Algorithms

E. Cantú-Paz

This article was submitted to
Genetic and Evolutionary Computation Conference 2001, San Francisco, CA, July 7-11, 2001

January 24, 2001

der of the relationships to be considered, but the algorithms proposed here can discover these relations as the search progresses. In addition, these evolutionary algorithms can handle constraints on the domain of each variable easily.

The paper is organized as follows. The next section briefly reviews evolutionary algorithms that use explicit models of the solutions to guide the search. Section 3 describes some existing supervised and unsupervised discretization algorithms and the corresponding evolutionary algorithms. Finally, section 4 summarizes the paper and offers some recommendations for future research.

2. Building Models to Guide the Search

The idea of building probabilistic models and using them to guide the search has been around for some time. As we would expect, the complexity of the models has increased over time as the methods of building models from data mature and more powerful computers become available. This section briefly reviews the previous work on evolutionary model-building algorithms. The interested reader should consult the reviews by Pelikan et al. (1999) and Larrañaga et al. (1999) for a more complete exposition of this area.

The simplest model-building evolutionary algorithms use a product of univariate and independent probability distributions as the model of solutions. Baluja (1994) introduced the Population-Based Incremental Learning (PBIL) algorithm. The PBIL uses a binary alphabet and its model is updated using a variation of the Hebbian learning rule used in neural networks. The compact GA (Harik, Lobo, & Goldberg, 1998) is another example of algorithms that use univariate models and operate on binary alphabets. The main difference with PBIL is that the learning rule mimics the behavior of a simple GA with a finite population size and uniform crossover. Closer to the algorithms presented here, the univariate marginal distribution algorithm (UMDA) estimates the probability densities from the individuals that survive the selection process (Mühlenbein, 1998).

Servais, de Jager, and Greene (1997) extended PBIL to discrete alphabets of higher cardinality and Sebag and Ducoulombier (1998) extended it to continuous variables. The model used by Sebag and Ducoulombier is a product of normal densities, and is similar to the model used by Rudlof and Köppen (1996) for their algorithm. The difference is in the rules used to adapt the parameters of the distributions in each case. Other algorithms that operate on continuous variables use more sophisticated univariate distributions. For example, Gallagher, Frean, and Downs (1999) used a mixture of normal distributions.

More sophisticated algorithms such as MIMIC (de Bonet, Isabell, & Viola, 1996) and the BMDA (Pelikan & Mühlebein, 1999) capture relationships between pairs of variables. The experimental evidence provided in the papers cited above suggests that, in many practical situations, simple uni- or bi-variate models are sufficient to find acceptable solutions. However, these results do not extend to more difficult problems that have interactions between more than two variables.

Other algorithms use richer models that allow relationships among arbitrary number of variables. Pelikan, Goldberg, and Cantú-Paz (1999), Etxeberria and Larrañaga (1999), and Mühlenbein and Mahnig (1999) introduced algorithms that learn Bayesian networks to represent the selected individuals. However, these algorithms consider only discrete variables. We could discretize the domain variables as a preprocessing step, but this would not be very efficient. For example, if the discretized variables are discretized into b bins, a node in a Bayesian network with $n-1$ parents would need b^n bins to represent the joint distribution of the n variables.

Larrañaga et al. (1999) proposed several algorithms for continuous domains. Their approach was progressive: first, they proposed a univariate algorithm that performs statistical tests to determine which of the candidate distributions best fits the data before determining the parameters; then they adapt the MIMIC bi-variate algorithm, and finally they propose a multivariate method based on learning a Gaussian network. The multivariate method begins with a complete Gaussian network, and performs all possible edge exclusion tests to identify conditional independences among the variables. If there are n variables, there are n^2 edges, so the algorithm may not be very efficient in high-dimensionality problems.

The more sophisticated model-building algorithms can consistently solve problems with complex interactions among many variables. However, constructing probabilistic networks from data is a costly operation. Bosman and Thierens (1999) proposed a general framework for evolutionary model-building algorithms, and showed how many of the existing algorithms can be instantiated from the framework. In addition, Bosman and Thierens discussed several ways to estimate the density of the selected solutions and examined the complexity of various algorithms. They showed that in the multivariate case, the complexities

Supervised and Unsupervised Discretization Methods for Evolutionary Algorithms

Erick Cantú-Paz CANTUPAZ@LLNL.GOV
Center for Applied Scientific Computing, Lawrence Livermore National Laboratory 7000 East Avenue, Livermore, CA 94550

Abstract

This paper introduces simple model-building evolutionary algorithms (EAs) that operate on continuous domains. The algorithms are based on supervised and unsupervised discretization methods that have been used as preprocessing steps in machine learning. The basic idea is to discretize the continuous variables and use the discretization as a simple model of the solutions under consideration. The model is then used to generate new solutions directly, instead of using the usual operators based on sexual recombination and mutation. The algorithms presented here have fewer parameters than traditional and other model-building EAs. We expect that the proposed algorithms that use multivariate models scale up better to the dimensionality of the problem than existing EAs.

1. Introduction

Recently there has been a growing interest in evolutionary algorithms (EAs) that build a probabilistic model of promising solutions and generate new solutions by sampling from the model. These algorithms seem like a promising path to construct reliable optimization algorithms that can solve difficult problems in reasonable times. However, most of these algorithms use discrete representations, which may not be natural to the problem at hand. To work on continuous domains, the users must discretize the problem's variables to a certain accuracy and decide on the alphabet and the encoding of the discretized variables.

This paper presents simple model-building evolutionary algorithms that work on continuous domains. The algorithms are based on supervised and unsupervised discretization methods that have been used in machine learning algorithms in the past. The basic idea behind the algorithms proposed here is to discretize the real variables and use the discretization as a simple model of the solutions under consideration. The model is used to generate new solutions directly; the evolutionary algorithm omits the usual operators based on sexual recombination and mutation. We distinguish between supervised discretization algorithms that use a class label to find intervals and unsupervised algorithms that do not use a label. In our case, the labels are determined by the selection method of the evolutionary algorithm.

Typically, discretization algorithms consider only one variable at a time, and some model building evolutionary algorithms that use simple univariate models have been shown to succeed in a variety of problems. However, it is well known that to succeed in problems where there are significant correlations between several variables, evolutionary algorithms must consider the related variables simultaneously or use exponential time. Therefore, we must consider multi-variate discretization methods. The discussion on this topic will focus on supervised discretization, and specifically on building decision trees that capture relationships between the most relevant variables, although we recognize that other supervised and unsupervised methods may also be adequate.

There are other evolutionary algorithms that operate on floating-point variables directly, but they do not use models to guide the search and may be limited to problems without significant interactions among many variables. However, these algorithms have been useful in many continuous domains and we can borrow some elements from them, such as a mutation operator that we use as a secondary search operator.

One of the benefits of the algorithms presented here is that they have fewer parameters than traditional and other model-building EAs. For example, since there have no recombination or mutation there is no need to choose these operators or their parameters. Other multivariate model-building EAs need to know the orvary from $O(n^2)$ to $O(n^3)$ depending on the algorithms used. This paper argues that in many cases simpler multivariate models may be sufficient to solve difficult problems reliably and quickly.

There have been other approaches to learn a model to guide the search of EAs. For example, Michalski (2000) introduced a system that induces rules that classify individuals into three groups depending on their fitness values. The rules are used to generate new individuals. Michalski's method can operate on continuous variables, but they have to be discretized first, so the algorithm includes a heuristic to modify the discretization levels as the search progresses.

3. Discretization-Based Evolutionary Algorithms

One way to characterize discretization methods in machine learning is to focus on whether they use information about the class label to determine the intervals. In an analogy to machine learning algorithms, the methods that use class information are called supervised, and those that do not use additional information are called unsupervised.

3.1 Unsupervised Methods

The simplest unsupervised discretization method is to divide the range of observed values into b bins of equal width. Although it is easy to implement, this method is very sensitive to outliers, and may not adequately represent the distribution of each variable. The user has to specify the number of bins b. This method computes the bin boundaries at

$$min_i + \frac{max_i - min_i}{b} k,$$

where min_i and max_i are the observed minimum and maximum values of the i-th variable, and $k = 1, ..., b$.

Another unsupervised method is to divide the range into b bins of equal frequency (so if there are n values, each bin would contain n/b elements). The equal frequency method is not susceptible to outliers, and the intervals would be closer to each other in regions where there are more elements and farther apart in sparsely-populated regions. This method can represent the distribution of each variable better than the equal width method. The user still has to decide the number of bins b.

The corresponding evolutionary algorithm is the same for these two unsupervised methods. The core loop is to perform selection to obtain n_s promising solutions, discretize each variable of these solutions independently, and for each variable generate n/b uniformly-distributed random numbers in each bin. These random numbers correspond to the values of the variable in the new population. Since the discretization captures the distribution of the *selected* individuals, we expect that the new individuals will have the same univariate marginal distributions than the selected individuals. This does not guarantee that the fitness values of the new individuals will be as good as the selected ones, unless the hypothesis of the variables being independent holds. We recognize that this hypothesis is not true in most practical problems, but the experimental results with other EAs that make the same hypothesis suggests that this method can be applied successfully even where there are dependencies, although it is probably limited to cases where the dependencies do not involve many variables.

This algorithm monotonically reduces the range of values for each variable after every generation, which is desired as we want to narrow the search to the most promising regions. But if the algorithm narrows the search too fast, it may impede a proper exploration of the search space, which might impact negatively the quality of the solutions found. A possible solution would be to use a large population to sample the search space better, but larger populations represent additional computational costs. Another solution would be to adopt a mechanism that slightly enlarges the range of values generated for the new population. One possibility would be to add additional bins with a few elements at the extremes of the discretized range, but this creates a few additional design decisions (e.g., what is the range of the additional bins and how many elements should we put in them?) that would make the algorithm more difficult to use. Another possibility is to adopt a mutation mechanism: with a low probability substitute the value of one of the new variables by a random value generated with a normal distribution with the mean and variance of the selected individuals.

Possibly the major problem with the methods described in this section (and other algorithms that depend on uni-variate models) is their inability to represent accurately disjoint regions of promising solutions. Consider the example depicted in figure 1. The small circles represent points that were selected by the algorithm because their observed fitness values are high. The algorithm would proceed to discretize the *entire* range of the selected individuals, which includes a large area of low performance. The equal-width bins method would be very sensitive to this problem since a large fraction of its bins would represent the range where no individuals were selected, and many individuals will be generated in the low-performing region. The equal-

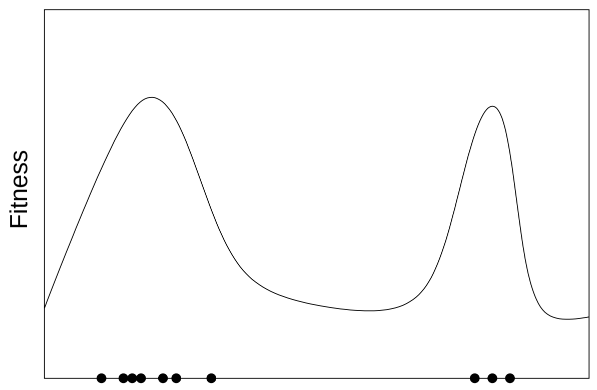

Figure 1. A fitness landscape with two disjoint high-performance regions.

frequency method would also generate individuals in the low-performance region, but many less than the equal-width algorithm.

Certainly, it may be useful to generate a few points in the regions where the observed performance of a variable is low, because it may be possible that the apparent low-performing region has not been sampled adequately and good solutions were missed. In addition, it might be possible to reach good results when the observed low variable is paired with other variables in regions that have not been sampled yet. However, in situations like those in the example above, where the low-performance region dominates the range of a variable, it may be wasteful to use these methods. The next section proposes methods that do not suffer from this problem.

3.2 Supervised Methods

The general idea of supervised discretization methods is to create intervals where all or most of the data instances have the same label and the labels are different across several intervals. Most supervised discretization methods have been used in combination with classification algorithms.

Supervised discretization methods assume that the data instances are labeled with the class to which they belong. The "supervised" name comes from an analogy where a "teacher" assigns labels to the instances and then evaluates the models created by the algorithm. In our case, the teacher is the selection method, and the individuals have binary labels corresponding to whether they were selected or not.

Holte (1993) proposed a simple algorithm that consists on sorting the observed values of a variable and greedily dividing the domain into bins that contain instances with the same label. Since this could lead to having one bin for each observed value, each bin is constrained to have a minimum number of elements.

Fayyad and Irani (1993) proposed a recursive method that finds intervals that minimize the class information entropy. We follow closely the notation of Dougherty et al. (1995). If we are given a set of instances S, a particular variable i, and a partition boundary T, the entropy of the partition induced by T is

$$Ent_i(S,T) = \frac{|S_l|}{|S|}Ent(S_l) + \frac{|S_r|}{|S|}Ent(S_r),$$

where S_l and S_r are the sets on the left and right respectively of the boundary T. The optimal boundary T^* that minimizes the entropy is chosen to partition the range, and the algorithm is applied recursively to the sets S_l and S_r. The stopping criteria is based on the minimum description length principle. The resulting intervals will be closer to each other in the regions with high entropy, and far apart in the uniform regions where the entropy is low.

In a sense, the supervised discretization methods create a rough model that predicts the class label based on the intervals. This model is likely to be too inaccurate to have a practical value as a classification algorithm, especially because it is based on a single feature, but it may be sufficient to guide the search of an evolutionary algorithm. After all, other EAs with univariate models perform well in some domains.

The corresponding evolutionary algorithm is the same for these two supervised methods, and is similar to the EA for unsupervised methods. The core loop is to apply the selection method to label the individuals, discretize each variable using all the individuals (not just the selected), and generate uniform random numbers only in the intervals that correspond to the selected individuals. Note that this algorithm can easily represent disjoint regions of promising solutions, and the ranges of promising solutions also decrease monotonically.

As we mentioned above, it might be useful to generate a few points in the regions of low performance. This observation might prompt some to modify the sampling procedure described above to generate a few points that correspond to the not-selected regions. However, this is not necessary, because all the popular probabilistic selection methods assign a non-zero probability of selection to all individuals, including those with low fitness. The exception are deterministic selection methods, such as truncation or $(\mu \overset{+}{,} \lambda)$, that select the top performing individuals.

The natural extension of these algorithms is to consider multivariate discretization. This, of course, is much more complex than discretizing one variable at a time, but we can continue to borrow from the machine learning field and use inductive learning algorithms to build a model of the individuals based on their labels.

For example, we can use decision trees as supervised discretization methods. Note that Fayyad and Irani's method can be regarded as building a tree on one variable. Kohavi and Sahami (1996) used C4.5 to discretize continuous variables as a preprocessing step in a classification task, but only applied it to a single variable. The idea is to use the binary splits of the tree as the threshold values for discretization. Kohavi and Sahami (1996) note that the main difference between this method and Fayyad and Irani's top-down method is the stopping criterion. C4.5 builds an entire tree in a top-down fashion that must be pruned (in a bottom-up fashion) before using the splits.

We may use decision trees to build a multi-variate model of the current solutions in a EA. The selection mechanism provides a label for every individual and the tree-building algorithm attempts to partition the solution space to minimize some impurity measure. The result is that the majority of the elements in a partition belong to the same class (i.e., selected or not-selected). The evolutionary algorithm can then use the tree to generate new individuals that correspond to the selected class.

Decision trees have been used with great success in classification tasks across a wide range of domains. Decision trees are popular for several reasons: they are fast to create, reasonably accurate in many domains, and easy to interpret by domain experts. The first two of these characteristics make decision trees very appealing to guide the search of EAs. For our purposes, the interpretability is not critical, but it may be interesting to observe how the trees change as the search progresses.

Another important characteristic that makes decision trees appealing is that they ignore variables that do not seem related to the class label. In classification problems this is a great advantage, because it makes fewer demands of preprocessing feature selection algorithms. In our case, ignoring irrelevant variables is also an advantage, because it permits the algorithm to focus on the variables that have the greatest influence on the fitness at a particular stage during the search. In a sense, this reduces the dimensionality of the problem over time. However, using trees introduces some difficulties on the generation of new individuals.

When the tree induction is complete, the path from the root to each leaf represents a conjunction of conditions that can be regarded as the antecedent of a rule. The label in each leaf indicates the class of the individual that satisfies every condition, and can be regarded as the consequent of the rule. It is not necessary that every rule (or path) involves all the variables that describe an individual, and it is possible that the same variable appears more than once in the same rule.

To generate the new individuals we use only the rules that correspond to the selected individuals. One rule is considered at a time, and one individual is generated from each rule. We could use the rules in a round-robin fashion to ensure that the rules get equal representation in the next population, or we could choose rules probabilistically based on the number or average fitness of the selected individuals that they represent. If a variable appears more than once in a rule and it defines an interval, then we simply generate uniform random numbers inside that interval. If a variable only appears once, then the missing extreme value can be the upper or lower limit of the variable (if it is known) or an arbitrary limit based on the current distribution of values (e.g., two times the standard deviation of the observed values in the current population.) Whatever we choose, if the variable appears once, some values would not be explored in the next generation, and the search would still being focused on promising solutions.

The most challenging difficulties occur if a variable does not appear in the rule, which is a case that we expect to happen frequently. It could be that the variable is irrelevant, so we could just generate a random number. But we have to consider that the variable may not be relevant at the current search stage, but it may have been relevant before, and actually it may have converged already to a unique value. To keep track of this situation, we complement the rules with a global template that keeps track of the ranges of the variables generated over the execution of the algorithm. The template would be updated every generation with the ranges of the new individuals. With this template, if a variable does not appear (or if only appears once) we can generate random numbers in the observed range. If a variable was relevant in an earlier stage of the search, and even if this variable has converged to a single value, the template would ensure that the new individuals have values that represent promising solutions. As before, mutations should be used occasionally.

Of course, we are not limited to use decision trees to build multivariate models of the current population, we could use any other learning algorithms that works on continuous domains, but unfortunately there are not many such algorithms. The approach followed here was to view discretization algorithms as learning a rough model of the current population. If we follow this approach we can foresee the application of other discretization algorithms such as the algorithm of Kozlov and Koller (1997) and vector quantization.

4. Conclusions

This paper proposed several evolutionary algorithms based on supervised and unsupervised discretization methods. First we discussed algorithms that discretize variables independently, which are similar to uni-variate model building EAs. We identified some potential shortcomings of these algorithms. Then we presented multivariate discretization based on decision tree induction.

The tree-based algorithms appear promising because tree induction methods avoid irrelevant features. This is equivalent to reducing the dimensionality of the search space dynamically, because those variables that are irrelevant at a particular stage of the search will be ignored. Because of this, we expect that these algorithms scale up better to the dimensionality of the search space than other model-building EAs.

In the future the algorithms will be tested on multiple continuous function optimization problems and compared against traditional and model-building EAs. There are multiple opportunities to extend the algorithms proposed here. For instance, unsupervised clustering algorithms can be used to discretize the domain, and they can be applied to single or multiple variables at a time. Likewise, there are other supervised discretization and machine learning algorithms that can be used to guide the search of an EA.

The algorithms presented here are easier to use than traditional and model-building EAs because they have fewer parameters. Although it is impossible to construct optimizers that perform well on all possible domains, the algorithms presented here represent a step toward the automatic, reliable, and accurate solution of difficult optimization problems.

Acknowledgments

UCRL-JC-142243. This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

References

Baluja, S. (1994). *Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning* (Tech. Rep. No. CMU-CS-94-163). Pittsburgh, PA: Carnegie Mellon University.

Bosman, P., & Thierens, D. (1999). *An algorithmic framework for density estimation based evolutionary algorithms.* Utrecht University Technical Report UU–CS–1999–46. ftp://ftp.cs.uu.nl/pub/RUU/CS/techreps/CS-1999/1999-46.ps.gz.

de Bonet, J., Isabell, C., & Viola, P. (1996). Mimic: Finding optima by estimating probability densities. In Jordan, M., Mozer, M., & Perrone, M. (Eds.), *Advances in Neural Information Processing Systems*, Volume 9. Cambridge, MA: MIT Press.

Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In Paredis, A., & Russell, S. (Eds.), *Machine Learning: Proceedings of the Twelfth International Conference* (pp. 194–202). San Francisco, CA: Morgan Kaufmann.

Etxeberria, R., & Larrañaga, P. (1999). Global optimization with Bayesian networks. In *II Symposium on Artificial Intelligence (CIMAF99).* (pp. 332–339).

Fayyad, U. M., & Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Thirdteenth International Joint Conference on Artificial Intelligence* (pp. 1022–1027). San Francisco, CA: Morgan Kaufmann.

Gallagher, M., Frean, M., & Downs, T. (1999). Real-valued evolutionary optimization using a flexible probability density estimator. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., & Smith, R. E. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference 1999: Volume 1* (pp. 840–846). San Francisco, CA: Morgan Kaufmann Publishers.

Harik, G. R., Lobo, F. G., & Goldberg, D. E. (1998). The compact genetic algorithm. In of Electrical, I., & Engineers, E. (Eds.), *Proceedings of 1998 IEEE Iternational Conference on Evolutionary Computation* (pp. 523–528). Piscataway, NJ: IEEE Service Center.

Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets.

Machine Learning, *11*, 63–90.

Kohavi, R., & Sahami, M. (1996). Error-based and entropy-based discretization of continuous features. In Simoudis, E., Han, J., & Fayyad, U. (Eds.), *Proceedings of the Second International Conference on Knowledge Discovery in Databases* (pp. 114–199). San Mateo, CA: AAAI Press.

Kozlov, A. V., & Koller, D. (1997). Nonuniform dynamic discretization in hybrid networks. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence* (pp. 314–325). San Mateo, CA: Morgan Kaufmann.

Larrañaga, P., Etxeberria, R., Lozano, J. A., & Peña, J. M. (1999). *Optimization by learning and simulation of Bayesian and Gaussian networks* (Tech Report No. EHU-KZAA-IK-4/99). Conostia-San Sebastian, Spain: University of the Basque Country.

Michalski, R. S. (2000). Learnable evolution process: Evolutionary process guided by machine learning. *Machine Learning*, *38*(1-2), 9–40.

Mühlenbein, H. (1998). The equation for the response to selection and its use for prediction. *Evolutionary Computation*, *5*(3), 303–346.

Mühlenbein, H., & Mahnig, T. (1999). FDA-A scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation*, *7*(4), 353–376.

Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (1999). BOA: The bayesian optimization algorithm. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., & Smith, R. E. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference 1999: Volume 1* (pp. 525–532). San Francisco, CA: Morgan Kaufmann Publishers.

Pelikan, M., Goldberg, D. E., & Lobo, F. (1999). *A survey of optimization by building and using probabilistic models* (IlliGAL Report No. 99018). Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Pelikan, M., & Mühlebein, H. (1999). The bivariate marginal distribution algorithm. In Roy, R., Furuhashi, T., & Chawdhry, P. K. (Eds.), *Advances in Soft Computing - Engineering Design and Manufacturing* (pp. 521–535). London: Springer-Verlag.

Rudlof, S., & Köppen, M. (1996). Stochastic hill climbing with learning by vectors of normal distribution. In *Proceedings of the First Online Workshop on Soft Computing (WSC1)* (pp. 60–70). Nagoya, Japan.

Sebag, M., & Ducoulombier, A. (1998). Extending population-based incremental learning to continuous search spaces. In Eiben, A. E., Bäck, T., Schoenauer, M., & Schwefel, H.-P. (Eds.), *Parallel Problem Solving from Nature, PPSN V* (pp. 418–427). Berlin: Springer-Verlag.

Servais, M., de Jager, G., & Greene, J. (1997). Function optimization using multiple-base population based incremental learning. In *Proceedings of the Eighth Annual South African Workshop on Pattern Recognition (PRASA '97)* (pp. 6–11).